Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea in Informatica

Algoritmi genetici per la ricerca di sottografi densi in reti temporali

Relatore: Prof. Italo Francesco Zoppis
Co-relatore: Prof. Riccardo Dondi

Relazione della prova finale di:
Riccardo Picozzi
Matricola 816974

Anno Accademico 2019-2020

Abstract

Vista l'enorme mole di dati che ogni minuto vengono scambiati in Rete, la loro analisi sta diventando un argomento centrale di studio e ricerca (network analysis and mining). Un campo relativamente nuovo è lo studio di reti temporali, grafi in cui nodi e archi cambiano nel tempo per rappresentare al meglio le variazioni sempre più frequenti nei sistemi dinamici complessi.

Il problema, trattato in questa tesi, riguarda la ricerca e l'identificazione di k gruppi coesivi (*comunità*) in una rete temporale. In particolare viene studiata una variante in cui gli intervalli temporali possono ammettere una sovrapposizione, limitata da un parametro α che vincola l'intersezione tra intervalli adiacenti.

Verrà presentato un metodo basato su algoritmi genetici e, con l'ausilio di un algoritmo greedy per il calcolo del sottografo denso, si otterà un metodo risolutivo per il problema. Verranno inoltre presentati i risultati sperimentali basati su reti reali di questa euristica, mostrandone performance e qualità delle soluzioni, al variare dei parametri come k, α, dimensione della popolazione e numero di generazioni.

Indice

Introduzione

I grafi (o reti) sono una struttura dati fondamentale per modellare le interazioni (archi) che si verificano tra entità (nodi). Ampiamente usate e analizzate, rivestono interesse in un'ampia gamma di campi applicativi, dall'informatica alla biologia, dalle scienze economiche a quelle sociali. Ogni grafo porta con sé un numero sempre crescente di strutture, modelli e algoritmi che ne consentono l'analisi: la disciplina chiamata teoria dei grafi. La sua origine può essere attribuita a Eulero grazie a un suo celebre scritto della metà del XVIII secolo intitolato *I sette ponti di Konisberg*, in cui per la prima volta i grafi sono presi in considerazione sotto forma di entità matematiche [16]. Procedendo con gli anni sono nati nuovi problemi che, con le relative soluzioni, hanno portato alla creazione di nuovi studi sull'analisi di grafi. E' nella seconda metà del XX secolo che si è ottenuto uno sviluppo significativo; l'introduzione dei computer ha fornito la possibilità di eseguire indagini sperimentali sui grafi, indirizzando lo studio di modelli e algoritmi anche al campo applicativo. A riguardo è facile citare il problema dei quattro colori: formulato a metà del XIX secolo è stato dimostrato oltre un secolo più tardi solo grazie all'avvento dei calcolatori, tramite i quali è stato possibile verificare, caso per caso, la veridicità dell'affermazione espressa nel problema [17]. I grafi rappresentano quindi una risorsa essenziale, oggi più che mai, nella modellazione e rappresentazione di dati merito anche delle numerose strategie collegate a esso.

Il mondo è sempre più indirizzato dall'enorme numero di dati che circolano nella rete, quindi, a partire dai colossi come Google o Microsoft, fino alle piccole aziende, tutti si affidano ai dati per prendere le loro decisioni e modificare le proprie politiche aziendali. Per questo scopo nasce la branca chiamata analisi dei dati (*Data analysis*): un processo che usa strumenti analitici e statistici per valutare i dati ed estrapolare le informazioni più utili e rilevanti. Queste informazioni verranno poi

utilizzate per conoscere le tendenze, i gusti o le relazioni degli utenti, per indirizzarli verso il loro obiettivo aziendale. Si parla di *network analysis* e *network mining* quando l'estrapolazione e l'analisi dei dati riguarda appunto le reti o i grafi. Questo campo ha un ruolo fondamentale nell'analisi di sistemi complessi, come ad esempio Facebook o Twitter, dato che con esso si può comprendere il comportamento e l'evoluzione dell'intero sistema.

In questo lavoro verrà trattato questo argomento, focalizzandosi in modo particolare sull'analisi di grafi temporali: reti particolari che risultano perfette per rappresentare le relazioni che cambiano nel tempo e, inoltre, si adattano in modo ottimale a diversi ambiti applicativi come le reti di comunicazione (ad es. lo scambio di telefonate tramite un operatore di telefonia mobile), le reti economiche (ad es. lo studio dei movimenti di carte di credito o criptovalute), i social network, le reti biologiche e tante altri. Proprio nell'ambito delle reti sociali si svolge il fulcro di questo lavoro, cioè la ricerca di comunità all'interno di essi. Esso prevede di trovare degli eventi significativi all'interno del grafo temporale che rappresenta la rete presa in considerazione, ottenendo così informazioni sulla sua organizzazione e rivelando relazioni nascoste che si sono verificate (e si verificano) all'interno.
I contenuti di questo lavoro sono divisi in tre sezioni principali. Il capitolo 1 descrive gli argomenti principali trattati in questa tesi presentando brevemente la loro storia, la loro evoluzione e il loro ruolo nello studio moderno dei grafi. Nel capitolo 2 sono presentate le nozioni base e le definizioni del problema e in seguito verrà proposto un algoritmo per la sua risoluzione basato sugli algoritmi genetici. Il capitolo 3 mostrerà alcuni risultati sperimentali dell'algoritmo proposto mostrandone le potenzialità. Le conclusioni verranno riportate nel quarto e ultimo capitolo.

Capitolo 1

Concetti base sullo studio di reti temporali

Lo studio di reti temporali è diventato di fondamentale importanza per l'analisi di sistemi dinamici come i social network o le reti di comunicazione. La velocità e la dinamicità con cui i dati mutano nella Rete e nel mondo ha fatto sì che questa disciplina ricoprisse un ruole sempre più di rilievo nell'ambito del network mining and analysis.

In questo capitolo verrà spiegata la sua evoluzione, introducendo la definizione di sottografi densi in grafi statici ed estendendo poi il concetto di ricerca di comunità alle reti temporali. Infine verrà introdotta una tipologia di algoritmi di ottimizzazione (euristica), gli algoritmi genetici, utilizzati per risolvere il problema.

1.1 Ricerca di sottografi densi

Uno degli aspetti più studiati nell'ambito del network mining and analysis è l'identificazione di comunità (*communities*) all'interno di una rete. Dato un grafo $G = (V, E)$ dove V rappresenta l'insieme di nodi ed E rappresenta l'insieme di archi presenti, ricercare una comunità significa trovare un sottoinsieme di nodi del grafo tali che presentano un forte legame che li collega tra loro. Le comunità rappresentano quindi un insieme di entità coesive, un gruppo di entità (i nodi del grafo in questo caso) significativi dell'intera rete. Per esempio consideriamo il grafo nella Figura **1** come se fosse un ipotetico scambio di telefonate (archi) tra diverse persone (nodi). Il problema ci chiede di trovare un gruppo di persone fortemente collegate tra loro

tali che rappresentano la rete; intuitivamente, si possono distinguere due comunità: l'insieme di nodi $\{a, b, c, d, e\}$ e l'insieme $\{g, h, i, l\}$, è necessario però un metodo matematico in grado di definire una comunità.

Originariamente l'idea proposta per ottenere questo risultato è stata quello di considerare le comunità come cricche (clique), cioè grafi completi definiti come segue:

Definizione 1. Un insieme di nodi S in un grafo $G = (V, E)$ è una cricca se ogni coppia di vertici $u, v \in S$ è adiacente.

Questo equivale a dire che un insieme di vertici del grafo è una cricca se e solo se ogni nodo è direttamente collegato a ogni altro vertice di questo insieme. Trovare una cricca massimale (una cricca con il maggior numero di vertici) in un grafo significava quindi trovare una comunità. Nel nostro esempio, considerando la cricca come modello per indicare una comunità, si può stabilire che l'insieme di nodi $V' = \{g, h, i, l\} \subset V$ formano una comunità poiché per ogni u, v in V' esiste un arco che li unisce.

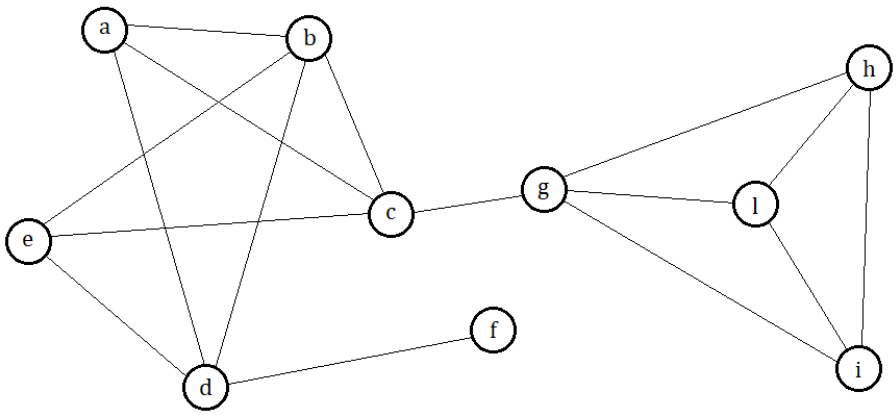

Figura 1.1: Esempio di grafo formato da 10 nodi e 16 archi.

Considerare le comunità come cricche è inadeguato, benchè esatto, per diversi fattori. Primo fra tutti perché questo modello è troppo stringente: considerare infatti che i nodi che compongono una comunità abbiano necessariamente un'interazione con tutti gli altri membri è restrittivo, dato che in una rete reale è molto difficile avere un

tale grado di connessione. Per esempio, il sottografo formato dai vertici $\{a, b, c, d, e\}$ mostra un grado di connessione molto alto, ma con il modello della cricca questo viene ignorato. Un secondo motivo riguarda la collezione dei dati perché spesso soggette a distorsioni (noise). Può accadere, infatti, che durante il campionamento qualche dato si perda o si modifichi accidentalmente facendo sì che una comunità reale non venga più identificata come tale, dato che la mancanza di un singolo arco potrebbe comprometterne la ricerca. Ammettiamo, ipoteticamente, che il ricercatore che ha fornito il grafo si fosse dimenticato l'arco (g, h), la comunità rilevata cambierebbe poiché l'insieme V' non formerebbe più una cricca per definizione.

Oltre a questi motivi, l'utilizzo di cricche come comunità comporta un'altra importante difficoltà: il problema della cricca è **NP-hard** [10]. Ciò significa che trovare la cricca massima è intrattabile in tempo polinomiale e difficile da approssimare.

Per far fronte a queste difficoltà si è pensato di definire dei rilassamenti (*clique relaxations*); un modello in cui una o più proprietà della cricca vengono "rilassate", come descritto da Pattillo in [1], in base a diversi aspetti della rete, come il grado dei vertici, il diametro, la densità o la connettività [7]. In questo modo si è in grado di ottenere delle definizioni alternative di comunità. Esempi dei problemi legati a questi rilassamenti, per citarne alcuni, sono: trovare il più grande *s-plex* in un grafo (degree-based relaxation), dove per s-plex si intende un sottoinsieme S dei vertici del grafo per cui ogni vertice del grafo indotto da S $G(S)$ ha grado almeno $|S| - s$ [2]; trovare il più grande *s-club* in un grafo (distance-based relaxation), dove un s-club è un sottoinsieme dei vertici in cui il grafo indotto da S $G(S)$ abbia diametro al massimo s [3].

Vista la difficoltà, e data l'importanza, i problemi computazionali legati ai rilassamenti della cricca hanno acquistato sempre più interesse negli ultimi anni, producendo molte ricerche che trattano i diversi aspetti del problema. Da questi sono emersi alcuni modelli basati su grafi densi o coesivi, come per esempio *s-cores* (sottografo che ha grado minimo s) [4] o *densest subgraphs* (sottografi con grado medio massimo). Questo lavoro si basa proprio sulla ricerca di comunità intese come sottografi densi.

Il problema della ricerca di sottografi densi in reti statiche è stato ampiamente studiato e analizzato in letteratura e il suo approccio è molto utilizzato tuttora

nell'analisi delle reti. Lo scopo è quello di trovare all'interno di un grafo un sottografo cha abbia la densità maggiore. Se consideriamo come comunità un sottografo più denso nell'esempio in figura, notiamo subito che il sottografo S_1 indotto dalla soluzione V' fornita precedentemente non è il migliore: infatti se calcoliamo la densità, intesa come il rapporto tra il numero di archi e il numero di vertici del sottografo, per S_1 otteniamo un valore inferiore (4 nodi, 6 archi) rispetto al sottografo S_2 indotto dai vertici $V'' = \{a, b, c, d, e\}$ (5 nodi, 8 archi).

Una menzione importante sull'argomento va sicuramente a Andrew Goldberg che, nel suo lavoro, fu tra i primi in grado di fornire una soluzione al problema in tempo polinomiale basandosi sul teorema max-flow min-cut. L'idea è stata quella di ricondurre il problema alla ricerca del sottografo con grado medio massimo, dato che la densità di un grafo è uguale alla metà del grado medio [5]. Il suo algoritmo però, data la sua complessità, nonostante sia rapido nel fornire una soluzione in reti di piccole dimensioni, risulta inefficace su grafi con un grande numero di nodi, perciò altri algoritmi di approssimazione sono stati proposti, in grado di fornire una soluzione molto vicina all'ottimo ma con tempi d'esecuzione di molto inferiori. Uno fra tutti l'algoritmo greedy di Charikar in grado di risolvere in tempo lineare il problema con un fattore di approssimazione pari a $\frac{1}{2}$ [6]. Il funzionamento di questo algoritmo è piuttosto semplice: dato un grafo (indiretto) calcola un insieme di suoi sottografi rimuovendo iterativamente il nodo di grado minore[1]. Il processo termina quando il grafo rimane con un singolo nodo. A questo punto, ottenuto l'insieme di sottografi, viene restituito quello con densità maggiore. Questo algoritmo è in grado di trovare un singolo sottografo denso all'interno della rete ma, come vedremo, il suo utilizzo risulta molto utile, per la sua rapidità e semplicità, in combinazione con altre procedure in grado di fornire buoni risultati in molte varianti del problema. Per esempio una ricerca sostanziosa è stata fatta sulla ricerca di un insieme di sottografi densi all'interno di un grafo dato [8][9]. Vista la sua rilevanza in questo ambito, e anche in questo lavoro, l'algoritmo di Charikar è risultato molto efficace nella ricerca di sottografi densi all'interno di reti temporali.

[1] il grado di un nodo è dato dal numero di vertici a cui esso è collegato direttamente (nodi adiacenti).

1.2 Reti temporali

I dati e le relazioni che li collegano variano nel tempo, a maggior ragione al giorno d'oggi in cui tutti siamo parte integrante del web. Basti pensare al numero di utenti registrati su Facebook o il numero di messaggi scambiati su Whatsapp ogni giorno per capire che ormai, in questi ambiti (e non solo), non si possono più modellare con grafi statici tutte le relazioni che in ogni secondo si formano e si sciolgono. E' quindi necessario aggiungere un nuovo parametro che indichi *quando* è avvenuta una certa interazione: il tempo. Proprio per questo scopo nascono le reti temporali, le quali possono essere definite informalmente come grafi che variano nel tempo. Si può pensare a esse come una sequenza di reti in cui ad ognuna è associata un'etichetta che cattura l'istante di tempo in cui una certa interazione è avvenuta. Una grande varietà di reti più o meno nuove sono state modellate come grafi temporali. Degli esempi validi possono essere le reti di informazione e comunicazione, i social network e le reti dei trasporti.

Vista la recente introduzione e nonostante i grafi statici siano stati studiati molto approfonditamente, la loro controparte temporale è ancora molto lontana dall'avere princìpi strutturali e algoritmi universalmente accettati. Alcune ricerche recenti indicano come molte proprietà e molti problemi riguardanti i grafi siano completamente diversi se a questi si aggiunge una nuova dimensione temporale. Per questi fattori questo ambito ha creato, negli ultimi anni, nuove sfide, nuovi problemi che non erano mai stati considerati prima portando a ricerche e soluzioni sempre più innovative.

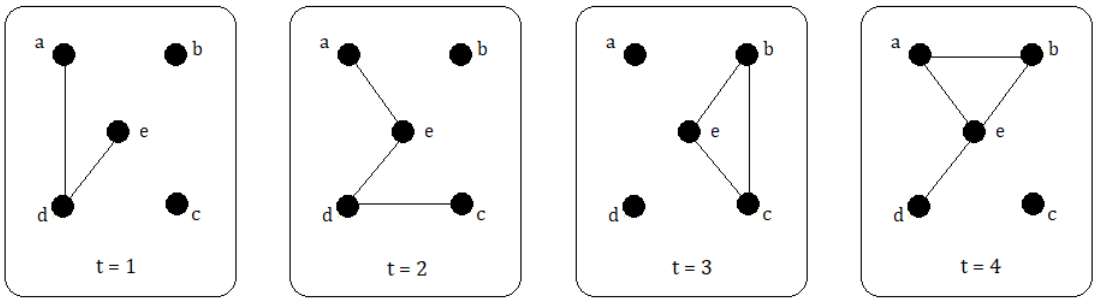

Figura 1.2: Esempio di grafo temporale formato da 5 nodi su 4 istanti di tempo differenti.

Esistono diversi modi in cui si può modellare un grafo temporale. Un metodo proposto in letteratura, e utilizzato in questo lavoro, è quello di considerare una rete temporale come una sequenza di grafi statici $G_1, G_2, \ldots, G_n$ ognuno dei quali etichettato da un intero, detto timestamp, che rappresenta l'istante in cui il grafo è attivo. In questo studio si assume che i vertici non cambino nel tempo.
Quindi, definiamo rete temporale un grafo $G = (V, T, E)$ dove V rappresenta l'insieme dei nodi, $T = [1, ..., t_{max}]$ rappresenta il dominio di tempo su cui è definito il grafo ed E rappresenta l'insieme di archi temporali definiti come una tripla (u, v, t) del sottoinsieme $V \times V \times T$, che equivale a dire che esiste un arco che collega u a v al tempo t. Un esempio di rete temporale si può vedere nella figura 2. Questa è formata dai nodi $V = \{a, b, c, d, e\}$ nel dominio di tempo $T = [1, 2, 3, 4]$. Al timestamp 1 sono presenti due interazioni (a, d) e (d, e), al tempo 2 (a, e), (d, e), (c, d) e così via.

Con l'aggiunta di una nuova dimensione, si è dovuto riformulare anche il problema della ricerca di sottografi densi, estendendo il problema alle reti temporali.

1.3 Ricerca di sottografi densi in reti temporali

Il problema di ricercare sottografi densi in un grafo temporale, soprattutto nel contesto dell'identificazione di episodi significativi all'interno delle attività online, ha prodotto diversi lavori e documenti a riguardo nonostante sia un ambito nato solo di recente, definendo anche delle linee guida e nuove idee per i lavori futuri.

Il problema, introdotto da Rozenshtein et al in [11], col nome di *k-Densest-Episodes*, prende in input un grafo temporale $G = (V, T, E)$ e ricerca $k \geq 1$ sottografi densi appartenenti a intervalli di tempo disgiunti, con l'obiettivo di ottenere una segmentazione del dominio di tempo T in cui ogni segmento (intervallo) evidenzi una comunità, portando così a una soluzione che mostra gli episodi più significativi che sono avvenuti durante il ciclo vitale della rete. Una possibile soluzione si ottiene partizionando il dominio di tempo in una sequenza di k intervalli disgiunti e, per ognuno di essi, se ne induce il sottografo calcolandone poi quello più denso. Un algoritmo che trova una soluzione in tempo polinomiale del problema *k-Densest-Episodes* viene fornito proprio da Rozenshtein et al.: questo algoritmo utilizza la programmazione dinamica per produrre una segmentazione del dominio di tempo e, in combinazione, l'algoritmo progettato da Goldberg per la ricerca di un sottografo più denso dell'intervallo dato. Nonostante la qualità delle soluzioni, esso risulta poco efficiente computazionalmente e, data la complessità di entrambi gli algoritmi, sia quello di programmazione dinamica che quello di Goldberg, non è utilizzabile in reti di grandi dimensioni.

Per far fronte a questo ostacolo è stato necessario cercare degli approcci più veloci, in cui si sacrifica la qualità della soluzione per ottenere una maggiore efficienza. [12] propone un algoritmo di approssimazione, KGAPPROX, che unisce un algoritmo che calcola un insieme di intervalli disgiunti sul dominio di tempo basato su un algoritmo di approssimazione di programmazione dinamica (ApproxDP), ad un algoritmo di approssimazione per il problema della ricerca del sottografo più denso (ApprDens). La qualità delle soluzioni e il tempo impiegato per trovarle, per entrambe gli algoritmi, dipendono dai paramentri e_1 e e_2, rispettivamente. Il fattore di approssimazione di KGAPPROX, come riportato in [12], è $2(1 + e_1)(1 + e_2)$; mentre il tempo computazionale è $O(\frac{k^2}{e_1 e_2^2}|T|mlog^2 n)$ dove k rappresenta il numero di intervalli, n il numero di nodi, m il numero di archi e T rappresenta il dominio di

tempo. Da questo si può notare che all'aumentare del valore dei parametri la velocità d'esecuzione aumenta mentre la qualità della soluzione diminuisce, e viceversa se si diminuiscono i parametri.

Da questo sono nati altri approcci, basati su euristiche, per risolvere il problema *k-Densest-Episodes*, con l'obiettivo di trovare un metodo più rapido per risolverlo. L'obiettivo delle euristiche è quello di produrre una soluzione a problemi computazionalmente pesanti, riconducendoli a problemi più semplici e veloci da calcolare ottenendo così un vantaggio in termini di prestazioni rispetto ad un metodo classico, sacrificandone però l'esattezza. Nei problemi NP-hard l'utilizzo di euristiche diventa fondamentale, data la difficoltà nel trovare una soluzione del problema in tempo polinomiale. Alcuni metodi offrono dei risultati molto vicini all'ottimo ma in tempi nettamente inferiori. Esempi di questo possono essere LSTDS (Local-Search Temporal Densest Subgraph) [13] e TDGA (Temporal Densest Genetic Algorithm) [8]. Il primo si basa sulla tecnica del *local-search*: viene computata una segmentazione iniziale basata sul numero di archi attivi e poi, iterativamente, tramite l'approccio local-search, si punta ad aumentare la densità totale dei k sottografi espandendo, uno per volta, gli intervalli della segmentazione. TDGA invece combina un algoritmo genetico, che si occupa della partizione del dominio di tempo della rete in intervalli disgiunti, e un algoritmo combinatorio (Charikar / Goldberg), il quale calcola il sottografo più denso per ogni intervallo della rete. Questo algoritmo porta a soluzioni molto vicine a quelle dell'algoritmo ottimo ma in tempi di molto inferiori.

Il lavoro di questa tesi prende spunto proprio da quest'ultimo algoritmo trattando una variante del problema in cui gli intervalli ammettono una sovrapposizione.

1.4 Algoritmi genetici

Gli algoritmi genetici sono un tipo di algoritmo di ottimizzazione, cioè usati per cercare la soluzione a un dato problema computazionale che massimizza o minimizza una funzione particolare. Essi rappresentano una branca del campo chiamato computazione evolutiva [14].

Nati tra la fine degli anni '50 e l'inizio degli anni '60, devono il loro nome al fatto che impiegano tecniche e meccanismi ispirati dall'evoluzione biologica, come la riproduzione, le mutazioni, la ricombinazione, la selezione naturale e la sopravvivenza del più forte.
Come nell'evoluzione biologica, molte funzioni e processi avvengono in modo casuale con la differenza, però, che in questi algoritmi il livello di casualità può essere impostato e controllato. La loro potenza ed efficienza deriva dal fatto che non necessitano di informazioni extra del problema dato, rendendoli quindi spesso migliori rispetto ad altri metodi di ottimizzazione che faticano nel gestire, per esempio, mancanze di continuità o linearità.

Data la sua fonte di ispirazione, molti termini usati negli algoritmi genetici derivano direttamente dalla biologia. I componenti base sono:

- i *cromosomi*: rappresentano una possibile soluzione del problema. Essi sono codificati come delle sequenze di valori (geni) che possono essere numeri binari o reali, tuple, alberi, ecc..;
- la *popolazione*: l'insieme di cromosomi su cui avviene il rimescolamento genetico, da cui si individua l'individuo migliore al termine dell'esecuzione dell'algoritmo;
- la *funzione di fitness*: è la funzione che l'algoritmo sta cercando di ottimizzare, qualifica quanto un cromosoma è adatto come soluzione;
- *operatore di selezione*: sceglie quanti e quali cromosomi far evolvere. Si può decidere di scegliere il migliore oppure decidere in base a una distribuzione di probabilità quale tra i cromosomi merita di passare alla generazione successiva;
- *operatore di crossover*: rimescola i cromosomi selezionati per creare la nuova popolazione della generazione successiva. Esistono varie tipologie di crossover con cui ricombinare i geni di un cromosoma, in base anche a come esso viene

rappresentato. Degli esempi possono essere il crossover a un punto (Figura 1.3), il crossover a due punti o il crossover uniforme;

- *operatore di mutazione*: data una certa probabilità muta in modo casuale un gene del cromosoma, dando così la possibilità di allontanarsi da un ottimo locale della funzione di fitness.

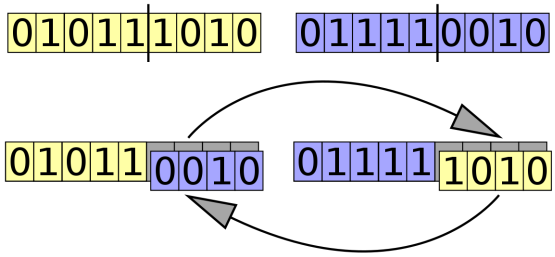

Figura 1.3: Esempio di crossover a un punto. Viene scelto in modo casuale un punto in entrambi i cromosomi genitori, e si formano due nuovi cromosomi formati dall'unione dei geni dei genitori come mostrato. Fonte: Wikipedia.

Ogni algoritmo genetico segue uno schema base con cui far evolvere le soluzioni durante la sua esecuzione. Questo schema è rappresentato in Figura 1.4. Per prima cosa viene inizializzata la popolazione in base ai parametri del problema: solitamente questo avviene in modo randomico, così da avere la maggior varietà di soluzioni disponibili. In seguito viene calcolato, per ogni inndividuo della popolazione, il valore di fitness: questo indicherà quanto il cromosoma sia meritevole di passare alla generazione successiva. La funzione di fitness viene creata ad hoc in base al problema che si sta affronatando. A questo punto la popolazione passa alla fase di rimescolamento genetico: vengono scelti i cromosomi migliori, vengono ricombinati tramite crossover e poi mutati secondo una certa probabilità. Le operazioni di selezione, mutazione e crossover sono iterate finché non viene soddisfatto il criterio di arresto. Questo criterio può considerarsi soddisfatto quando la soluzione raggiunge un valore ottimale (stabilito a priori) oppure quando è stato iterato un numero prefissato di volte (generazioni). Una volta soddisfatto il criterio d'arresto, si individua il cromosoma migliore della popolazione finale, il quale diventerà la soluzione migliore del problema.

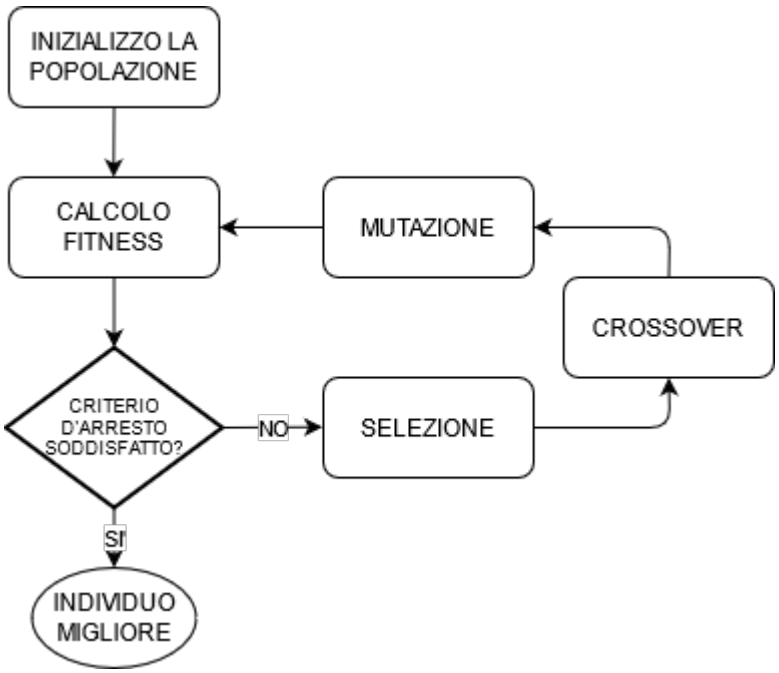

Figura 1.4: Schema di funzionamento di un algoritmo genetico.

Gli algoritmi genetici sono tuttora utilizzati in una grande varietà di applicazioni come la programmazione automatica e il machine learning. Risultano anche adatti alla modellazione di fenomeni in economia, biologia, ecologia, sistemi sociali e molti altri sistemi, nonché in molti i problemi di ottimizzazione. [15] offre un ottimo approfondimento a riguardo.

Capitolo 2

k-Densest-Overlapping-Episodes

In questo studio viene affrontata una variante del problema *k-Densest-Episodes* introdotto da Rozenshtein et al (come descritto in precedenza), ossia la ricerca di sottografi densi in reti temporali. Fu lo stesso autore nel suo lavoro [12] a offrire degli spunti sulle possibili varianti del problema da lui risolto. Uno di questi fu proprio la ricerca di sottografi densi in reti temporali dove veniva ammessa una sovrapposizione tra gli intervalli in cui è diviso il dominio di tempo. Da questo nasce il problema *k-Densest-Overlapping-Episodes* trattato in questo progetto in cui si proverà a stabilire se e quanto influisca l'aggiunta di questa condizione, rispetto alla qualità delle soluzioni ritornate.

Verranno ora introdotti il problema e le definizioni utili alla sua comprensione, ed in seguito verrà descritto il funzionamento dell'algoritmo creato per risolvere tale problema.

2.1 Problema

Il problema richiede di risolvere il problema della ricerca di sottografi densi in una rete temporale in cui è possibile avere una sovrapposizione tra interalli di tempo adiacenti. Lo scopo è verificare che, ammettendo una certa sovrapposizione tra gli intervalli, si è in grado di aumentare la qualità di una soluzione senza sacrificarne l'efficienza. Per stabilire il livello di sovrapposizione è stato introdotto un nuovo parametro α che lo limita. Il problema può essere definito nel seguente modo:

Definizione 2. Dato un grafo temporale $G = (V, T, E)$, trovare un insieme di k episodi di densità massima su intervalli di tempo distinti con un limite, regolato da α, nella sovrapposizione. I vincoli sulla sovrapposizione sono espressi dalle due disuguaglianze $\frac{|I_i \cap I_{i+1}|}{|I_i|} \leq \alpha$ e $\frac{|I_i \cap I_{i+1}|}{|I_{i+1}|} \leq \alpha$, con $0 \leq \alpha < 1$.

I due rapporti mostrano quanto l'intervallo si sovrappone al suo precedente e al suo successivo, indicando la sua percentuale di sovrapposizione. I vincoli ci dicono quindi la percentuale di sovrapposizione massima in base ad α che ogni intervallo può avere. Questo valore è pensato per far sì che l'intersezione non sia troppo ampia evitando così la completa inclusione di un intervallo in quelli adiacenti, rischiando, altrimenti, di ottenere una soluzione in cui due intervalli offrono come risultato lo stesso sottografo denso, perdendo così di vista l'obiettivo della ricerca. Infatti, se ammettessimo per esempio $\alpha = 1$, significherebbe che l'intervallo preso in considerazione sia formato esattamente dalla sua intersezione, e quindi interamente compreso nell'intervallo successivo.

L'algoritmo di risoluzione per questo problema è composto da due parti: la prima parte prevede un algoritmo genetico che partiziona il dominio di tempo discreto in k sezioni (inizialmente casuali) e opera su di essi per cercare di massimizzare la soluzione tramite il rimescolamento genetico, aumentando e diminuendo l'ampiezza degli intervalli in base al risultato della funzione di fitness; la seconda parte è l'algoritmo combinatorio che calcola il sottografo più denso in un grafo statico (fitness), il quale si ottiene considerando il grafo attivo in quell'intervallo di tempo. La scelta per il calcolo della densità, e del relativo sottografo più denso, è ricaduta sull'algoritmo greedy di Charikar vista l'efficienza e la qualità delle soluzioni ritornate.

Verranno ora presentate delle definizioni utili per comprendere al meglio le variabili del problema e in seguito verrà spiegato il funzionamento dell'algoritmo.

2.2 Definizioni

Ogni grafo temporale è definito su un dominio di tempo discreto $T = [t_1, t_2, \ldots, t_n]$, dove ogni t_i, detto *timestamp*, è un intero appartenente all'insieme $\{1, \ldots, n\}$ tale che per ogni $1 \leq i \leq n-1$, $t_i < t_{i+1}$.

Un intervallo in T è definito come $I = [t_i, t_j]$ in cui $t_i, t_j \in T$ con $1 \leq i, j \leq n$ e $t_i < t_j$. Da questi si ottiene un insieme $S = \{I_1, I_2, ..., I_k\}$ di intervalli sovrapposti dove, per ogni intervallo $I_i = [t_{i_1}, t_{i_2}]$ e $I_j = [t_{j_1}, t_{j_2}]$ in S, $1 \leq i, j \leq k$, $t_{i_1} < t_{i_2}$ ed è sempre rispettato i vincoli su α $\frac{|I_i \cap I_{i+1}|}{|I_i|} \leq \alpha$ e $\frac{|I_i \cap I_{i+1}|}{|I_{i+1}|} \leq \alpha$.

Tutti i grafi temporali considerati in questo lavoro sono indiretti. Inoltre si assume che i nodi della rete non cambino tra un istante e l'altro di tempo mentre possono variare gli archi. Per ogni intervallo vengono considerati solo i nodi che hanno almeno un arco incidente all'interno dell'intervallo, altrimenti non vengono presi in considerazione.

Possiamo ora definire un grafo temporale $G = (V, T, E)$ dove V rappresenta l'insieme dei nodi, T il dominio di tempo ed E l'insieme di archi temporali nel sottoinsieme $V \times V \times T$. Un arco è quindi una tripla (u, v, t_i) dove $u,\ v$ sono due nodi della rete e t_i un timestamp con $1 \leq i \leq n$: questo equivale a dire che esiste al tempo t_i un arco che collega u a v. Il suddetto arco è chiamato arco attivo. Prendendo in considerazione una rete temporale $G = (V, T, E)$ e un intervallo I in T, il grafo indotto $G_I = (V_I, E_I)$ formato da tutti gli archi attivi in un timestamp dell'intervallo e dall'insieme di vertici che collegano, è detto grafo attivo. Prendiamo per esempio il grafo temporale mostrato in Figura 1.2: se assumiamo un intervallo $I' = [1, 2]$ il grafo attivo corrispondente $G_{I'}$ sarebbe formato dai verici $V' = \{a, c, d, e\}$ e dall'insieme di archi attivi $E' = \{(a, c), (a, d), (c, d), (d, e)\}$.

Un episodio è definito come il sottografo $G'_I = (V'_I, E'_I)$ di un grafo attivo $G_I = (V_I, E_I)$ tale che $V'_I \subset V_I$ e $E'_I \subset E_I \times (V'_I \times V'_I)$ in cui la densità è massima. La densità è definita come il rapporto tra il numero di archi e il numero di vertici del sottografo:

$$dens(G_I) = \frac{|E|}{|V|}.$$

Considerando ancora una volta il grafo in Figura 1.2, un episodio nell'intervallo $I = [3, 4]$, può essere il sottografo G'_I formato dai vertici $V'_I = \{a, b, c, e\}$ e dagli

archi $E'_I = \{(a,b),(a,e),(b,c),(b,e),(c,e)\}$ con densità pari a $\frac{5}{4}$. Si può notare che nella soluzione non è stato considerato il nodo d e l'arco (d,e), nonostante siano presenti nel grafo attivo, poichè ne avrebbero diminuito la qualità ($\frac{6}{5}$).

E' possibile ora definire il problema che ha come obiettivo la ricerca di eventi significativi nella rete temporale.

k-Densest-Overlapping-Episodes

Input: Un grafo temporale G = (V,T,E), un intero k e un valore α, con $0 \leq \alpha < 1$.

Output: Un insieme P = $[G'_{I_1}, ..., G'_{I_k}]$ di k episodi sovrapposti in G tali che $\frac{|I_i \cap I_{i+1}|}{|I_i|} \leq \alpha$, $\frac{|I_i \cap I_{i+1}|}{|I_{i+1}|} \leq \alpha$, per ogni $i < k$, e $\sum_{j=1}^{k} dens(G'_{I_j})$ è massima.

2.3 Algoritmo genetico

Dato un grafo temporale $G = (V, T, E)$ su un dominio di tempo $T = [t_1, ..., t_{max}]$, un intero k che indica il numero di intervalli e un valore reale α, con $0 \leq \alpha < 1$ che limita la sovrapposizione tra intervalli adiacenti, rappresentiamo un insieme di episodi come un cromosoma di k elementi

$$C = [(c_1, \delta_1), ... , (c_{k-1}, \delta_{k-1}), t_{max}]$$

formato da $k-1$ coppie ordinate di interi e dal valore t_{max} che rappresenta l'ultimo timestamp del grafo temporale. Ogni coppia ordinata è composta da:

- c_i (*point*) un intero che rappresenta il timestamp iniziale dell'intervallo $i+1$. Gli elementi sono disposti in modo ordinato tra t_1 e t_{max}; per ogni i in k: $c_i < c_{i+1}$;
- δ_i (*overlap*) è un intero che rappresenta la sovrapposizione tra l'intervallo corrente e il precedente. $\delta_i = |I_i \cap I_{i-1}|$.

Gli intervalli saranno costruiti nel seguente modo: per ogni i tra 2 e $k-1$ l'i-esimo intervallo sarà $I_i = [c_{i-1}, c_i + \delta_i$ - 1].
Il primo intervallo inizierà sempre da 1. Quindi $I_1 = [1, c_1 + \delta_1$ - 1].
L'ultimo intervallo terminerà sempre in t_{max}.

Esempio:
$I_1 = [1, c_1 + \delta_1$ - 1]
$I_3 = [c_2, c_3 + \delta_3$ - 1]
$I_k = [c_{k-1}, c_k]$

A ogni cromosoma è associato un vettore f_c che consiste di k valori: l'i-esimo valore di f_c, con $1 \leq i \leq k$, rappresenta la densità massima del sottografo temporale nell'i-esimo intervallo di C. Questo valore è calcolato tramite l'algoritmo di Charikar.

2.3.1 Intervallo di ammissibilità dell'overlap δ

Come detto in precedenza, dato un grafo temporale G, un intero k e un valore α compreso tra 0 e 1, bisogna fare in modo che le condizioni

1. $\frac{|I_i \cap I_{i+1}|}{I_i} \leq \alpha$

2. $\frac{|I_i \cap I_{i+1}|}{I_{i+1}} \leq \alpha$

siano sempre rispettate.

Per ogni intervallo i del cromosoma consideriamo l'intersezione $|I_i \cap I_{i+1}|$ come δ_i (per comodità) e come x_i la dimensione dell'i-esimo intervallo privo della sovrapposizione $x = |I_i \setminus I_i \cap I_{i+1}|$. Quindi, $x_i = c_i - c_{i-1}$. Ad esempio dato C = [(10, δ_1), (20, δ_2), (35, δ_3), 50] per i=2, x_2 sarà $c_2 - c_1 = 10$.

Una volta riscritto il vincolo 1 si ottiene $\frac{\delta_i}{x_i + \delta_i} \leq \alpha$. A questo punto è facile calcolare il valore di δ in funzione di α e x (valori noti) calcolandone la funione inversa, ottenendo:

$$\delta_i \leq \frac{\alpha x}{1-\alpha} \quad (1)$$

Procedento nello stesso modo, riscrivo anche il vincolo 2 ottenendo il nuovo vincolo $\frac{\delta_i}{x_{i+1} + \delta_{i+1}} \leq \alpha$. In questo modo è possibile scrivere la disuguaglianza in funzione di δ_i:

$$\delta_i \leq \alpha * (x_{i+1} + \delta_{i+1}) \ (2)$$

Ora il problema richiede che entrambe le disuguaglianze valgano contemporaneamente per ogni intervallo. Data la disequazione (1) posso rapidamente calcolare un valore intero, per ogni intervallo del cromosoma, tale che funga da limite superiore per δ_i: $\delta_i = \frac{\alpha x_i}{1-\alpha}$. Per quanto riguarda la seconda disuguaglianza è necessario fare un ulteriore ragionamento. In fase di inizializzazione del cromosoma, per ottenere un limite superiore per δ_i è necessario prima calcolare il valore di δ_{i+1} poichè esso è incognito. Per ovviare questo problema si è pensato di inizializzare le sovrapposizioni del cromosoma partendo dall'ultimo overlap e procedere a ritroso; in questo modo, al primo passaggio, si può ottenere facilmente un limite superiore per δ_{k-1} poichè non esistono ulteriori sovrapposizioni successive. Una volta calcolato, passo al delta precedente in cui tutte le variabili sono note e così via fino a completare il cromosoma.

Fatte queste considerazioni è possibile stabilire l'intervallo di ammissibilita per tutte le sovrapposizioni del cromosoma facendo in modo che i vincoli su α siano sempre rispettati. Il valore massimo che δ_i può avere sarà dato dal minimo tra le due condizioni, ottenendo quindi il seguente range di ammissibilità:

$$[\ 0,\ min\{(1),(2)\}\ ]$$

dal quale è possibile scegliere un valore, in modo casuale, da assegnare a δ, tale per cui esso sia ammissibile.

Queste saranno anche condizioni necessarie durante il rimescolamento genetico dato che si potrebbe incorrere in situazioni in cui δ sia troppo ampio e quindi modificarlo di conseguenza.

Va tenuto anche conto che un intervallo non può superare il valore di t_{max} quindi va posta una condizione che blocchi questa evenienza, cioè:

$$c_i + \delta_i < t_{max} \text{ per ogni } i \text{ con } i < k$$

Se questa condizione non è rispettata bisogna modificare il range di ammissibilità di δ in [0, t_{max} - c_i - 1] e scegliere un valore di overlap all'interno di esso.

2.3.2 Funzionamento dell'algoritmo

Definiamo una popolazione iniziale di h cromosomi. Per prima cosa vengono inizializzati i point c_i tralasciando momentaneamente gli overlap δ_i corrispondenti. Per ogni elemento del cromosoma il valore δ viene inizializzato a zero ottenendo un cromosoma come il seguente:

$$C_i = [(c_1, 0), (c_2, 0), ..., (c_{k-1}, 0), t_{max}].$$

I valori c_i degli h cromosomi rappresentanti la popolazione saranno creati secondo queste regole:

- Un cromosoma sarà formato da un insieme di intervalli che partiziona T = [t_1, ..., t_{max}] in modo che ciascun intervallo contenga uguale numero di archi.
- Un cromosoma che partiziona il dominio di tempo in k intervalli di stessa dimensione.
- un insieme di h - 2 cromosomi, ognuno definito partizionando il dominio di tempo in modo casuale.

A questo punto si passa a calcolare gli overlap facendo in modo che i vincoli su α siano sempre rispettati. A questo proposito ci viene in aiuto la formula (calcolata nel paragrafo precedente) per trovare un valore δ_i che rispetti sempre i vincoli. Quindi dato l'intervallo di ammissibilità dell'overlap, per ogni i tra 1 e k - 1, scelgo casulamente un valore contenuto al suo interno con cui inizializzare l'i-esimo overlap. Una volta scelto un δ adeguato per ogni elemento del cromosoma abbiamo ottenuto una popolazione iniziale di h cromosomi $C = [(c_1, \delta_1), (c_2, \delta_2), ..., (c_{k-1}, \delta_{k-1}), t_{max}]$ da cui cominciare il processo di rimescolamento genetico.

Tutti i cromosomi sono valutati tramite la funzione di fitness che consiste nella somma di tutti i valori del vettore f_c e si evolvono tramite tre funzioni principali: *mutazione*, *crossover* ed *elitist selection*.

- **Elitist Selection**: per evitare che la densità della soluzione diminuisca tra due generazioni, un sottoinsieme della popolazione (i cromosomi con fitness più alta) viene copiato direttamente nella popolazione successiva. In questo lavoro vengono scelti gli $h/3$ cromosomi migliori.
- **Crossover**: viene effettuato un crossover a un punto: dato un intero p random compreso tra 1 e k-2 consideriamo due cromosomi nella popolazione C_1 e C_2 (*genitori*) e da cui deriviamo due cromosomi C_1' e C_2' (*figli*). I cromosomi genitori sono scelti tramite una *selezione a roulette* in cui la probabilità che un cromosoma della popolazione venga scelto è direttamente proporzionale al valore restituito dalla funzione di fitness. Per esempio, con k=3, se la funzione di fitness restituisce come totale 100, e il singolo cromosoma ha totalizzato 20, significa che la probabilità che questo venga scelto come genitore sarà $\frac{20}{100}$, cioè il 20%. Una volta selezionati i genitori possiamo derivare due cromosomi figli nel seguente modo:
  - C_1' contiene le prime p coppie di C_1 e le seguenti $k - p$ coppie di C_2
  - C_2' contiene le prime p coppie di C_2 e le seguenti $k - p$ coppie di C_1

  A questo punto si riordinano i due cromosomi figli in ordine crescente secondo i point controllando che nessun point sia ripetuto (in questo caso si sostituisce uno dei point con un'altro scelto a caso nell'intervallo [1, t_{max}] tale che non sia presente nel cromosoma) e si verifica che tutti gli overlap si trovino all'interno

della regione di ammissibilità. Se si incontra un overlap che non rispetta tale vincolo, si può agire in due modi:

1. Decremento δ fino a renderlo accettabile. Per esempio se l'i-esimo intervallo di ammissibilità calcolato è [1, 20] e l'overlap dell'i-esimo point è 25, decremento δ_i fino a renderlo uguale a 20.
2. Scelgo un nuovo δ in modo casuale nell'intervallo di ammissibilità dell'overlap.

In questo progetto si utilizza il primo caso, massimizzando le sovrapposizioni, al fine di trovare delle soluzioni migliori.

Se le condizioni sono soddisfatte per ogni coppia del cromosoma dei figli, essi sono copiati nella popolazione della generazione seguente.

- **Mutazione**: ogni coppia (tranne l'ultimo elemento) del cromosoma può mutare in modo random con una probalbilità $\frac{1}{k}$. Nel caso l'i-esimo elemento muti viene scelto random c_i nell'intervallo $[c_{i-1} + 1, c_{i+1} - 1]$ il nuovo valore che prenderà il posto del point mutato. L'intervallo è scelto per evitare che il nuovo elemento finisca completamente incluso in un altro intervallo adiacente. A questo punto scelgo un overlap δ_i casualmente all'interno del range di ammissibilità, come spiegato in precedenza, ottenendo una nuova coppia che sostituirà la coppia mutata.

La procedura viene iterata per g generazioni al termine della quale si ottiene una possibile soluzione del problema (criterio d'arresto).

Capitolo 3

Analisi sperimentali

In questa parte del lavoro verrà testato l'algoritmo che risolve il problema *k-Densest-Overlapping-Episodes* appena descritto mostrando la qualità delle soluzioni e il tempo di esecuzione, al variare dei parametri k, α, dimensione della popolazione e numero di generazioni. Verranno poi analizzati i dati, comparando la densità e l'efficienza risultanti al variare dei parametri stabiliti.

3.1 Ambiente di test

I test sono stati eseguiti su un PC Asus (Windows 10 Home) con un processore i7-7700 da 2.8GHz e 8GB di RAM DDR4. Per avere dei dati statisticamente significativi lo script è stato eseguito in modo indipendente 50 volte per ogni combinazione dei parametri. I risultati sono mostrati nelle Tabelle 3.1, 3.2, 3.3 dove vengono mostrate le medie della densità del sottografo più denso e del tempo d'esecuzione delle 50 esecuzioni. Il programma è stato implementato nel linguaggio Python.

3.2 Parametri

La scelta dei parametri è fondamentale per valutare l'efficienza e l'efficacia dell'algoritmo, poichè variando anche uno di essi si possono ottenere risultati drasticamente diversi. Sono state testate diverse combinazioni di parametri, quelle utilizzate sono state scelte poichè fornivano i risultati migliori.

Lo script è stato eseguito in tutte le combinazioni possibili variando dimensione della popolazione (h) e numero di generazioni (g) :

- $h = 10,\ \ g = 5$
- $h = 100,\ \ g = 5$
- $h = 10,\ \ g = 10$
- $h = 100,\ \ g = 10$

e modificando inoltre k e α per ogni valore negli insiemi :

- $k = [5,\ 10,\ 20]$
- $\alpha = [0.05,\ 0.1,\ 0.2]$.

La dimensione della popolazione e il numero di generazioni influiscono molto sull'efficienza dell'algoritmo soprattuto in reti di grandi dimensioni, per questo si è scelto di limitare le generazioni a 10 e la popolazione a 100, così da evitare un dispendio di risorse troppo elevato; sono state testate alcune combinazioni con un numero maggiore di cromosomi e di generazioni ottenendo risultati del tutto simili a quelli ottenuti dalla combinazione più lenta ($g = 10,\ \ h = 100$). Inoltre questi parametri sono gli stessi usati in [8] dove si ricercano i sottografi più densi in intervalli disgiunti, in modo da avere un confronto diretto con i risultati ottenuti in questo lavoro, in cui, invece, è ammessa la sovrapposizione.

Per quanto rigurada α sono stati utilizzati dei valori piccoli così da evitare sovrapposizioni eccessive che avrebbero potuto portare a delle soluzioni poco significative, e non in linea con l'obiettivo di ricerca. Questi valori sono stati scelti per mettere in luce se e quanto l'aggiunta di una sovrapposizione porti a un miglioramento nella soluzione.

3.3 Dataset

Per analizzare le performance dell'algoritmo sono stati utilizzati quattro differenti dataset reali pubblici ricavati da piattaforme social (Facebook e Twitter) e scambi di e-mail (Students ed Enron). I dataset, di cui vengono riportati il numero di interazioni, il numero di nodi e la dimensione del dominio di tempo, sono i seguenti:

1. *Students*: 10000 archi, 889 nodi, 1000 timestamps.
2. *Enron*: 6245 archi, 1143 nodi, 815 timestamps.
3. *Facebook*: 10000 archi, 4117 nodi, 9984 timestamps.
4. *Twitter*: 11868 archi, 4605 nodi, 9986 timestamps.

Il primo dataset, *Students*, raccoglie gli scambi di messaggi tra studenti dell'Università della California tra il 27/06/2004 e il 26/10/2004. In questa rete i nodi rappresentano gli studenti univeristari, mentre gli archi rappresentano i messaggi scambiati [1]. Il secondo dataset, *Enron*, contiene le e-mail inviate e ricevute (archi) tra il 01/01/1980 e il 13/02/2002 dai manager e dirigenti (nodi) dell'omonima multinazionale statunitense, rese pubbliche dopo l'indagine diretta dal Federal Energy Regulatory Commission. Il terzo dataset, *Facebook*, rappresenta una porzione dell'attività social della comunità regionale di New Orleans tra il 09/05/2006 e il 20/08/2006 in cui i nodi rappresentano gli utenti Facebook mentre le interazioni rappresentano i post pubblicati sulla bacheca di un altro utente. Il quarto e ultimo dataset, *Twitter*, raccoglie le interazioni tra utenti Twitter di Helsinky scambiate tra il 31/07/2010 e il 31/10/2010. Le interazioni di questa rete rappresentano i tweet pubblicati da un utente (nodi) in cui viene menzionato un altro utente.

3.4 Risultati

I risultati ottenuti sono frutto di 50 esecuzioni indipendenti dell'algoritmo per ogni combinazione dei parametri k, α, numero di generazioni e dimensione della popolazione.

Nelle tabelle seguenti vengono mostrati i valori medi di densità del sottografo più denso e i valori medi del tempo impiegato (espresso in secondi), al variare dei parametri. Nella Tabella 3.1 sono riportati i valori dell'esecuzione dell'algoritmo che risolve il problema per $k = 5$, la Tabella 3.2 mostra i valori per $k = 10$, mentre nella Tabella 3.3 i valori per $k = 20$.

Nella Figura 3.1 è possibile vedere un esempio delle comunità, intese come sottografi più densi, rilevate dall'algoritmo. La figura mostra l'esecuzione per $k = 5$

[1] Dato che in questi test vengono utilizzati solo grafi indiretti, la direzione dei messaggi è stata ignorata

ed $\alpha = 0.1$, sul dataset *Enron*. Il processo di rimescolamento genetico è iterato per 4 generazioni su una popolazione di 10 individui passando da una densità pari a 41.34 nella prima generazione a 42.82 nella quarta, con un incremento del 3.58%. Nella figura vengono riportati i sottografi più densi corrispondenti ai 5 intervalli in cui è suddiviso l'individuo migliore di ogni generazione.

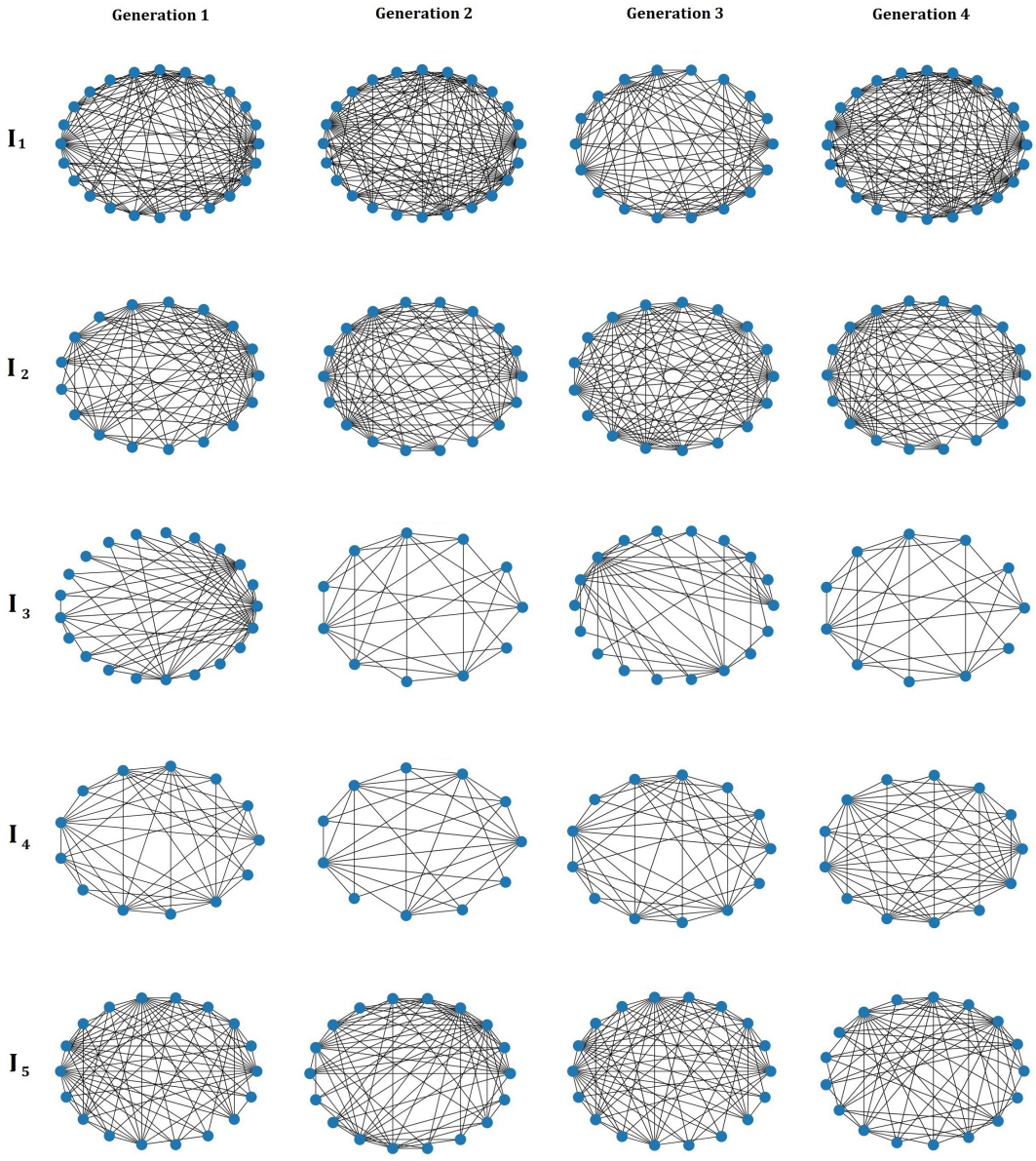

Figura 3.1: Esempio di esecuzione dell'algoritmo sul dataset *Enron* con parametri $k = 5,\ \alpha = 0.1,\ h = 10,\ g = 4$.

Come si può notare in tutte e tre le tabelle, i valori medi della densità al variare dei parametri g e h non mostrtano grandi differenze. Anche se si ottiene una soluzione migliore in tutti i casi, l'incremento può considerarsi solo marginale: se compariamo i valori di densità nel caso migliore ($g = 10$ & $h = 100$) con quelli nel caso peggiore ($g = 5$ & $h = 10$) in media si ottiene un miglioramento del 2.31%, con un massimo del $5,21\%$ (dataset *Facebook* con $k = 5$ e $\alpha = 0.1$) e un minimo dello 0.37% (dataset *Students* con $k = 20$ e $\alpha = 0.2$). Mentre la qualità della soluzione aumenta, le prestazioni in termini di tempo subiscono un brusco rallentamento: se poniamo a confronto sempre i due casi agli estremi, cioè $g = 5$ & $h = 10$ e $g = 10$ & $h = 100$, si può notare come quest'ultimo sia nettamente più lento. Per la precisione, l'esecuzione dell'algoritmo nel caso peggiore impiega almeno 16.83 volte il tempo impiegato dalla controparte più veloce, con un massimo di 21.21 volte e una media di 19.73. Il parametro che più influisce sui tempi d'esecuzione è sicuramente la dimensione della popolazione. Se assumiamo come numero di generazioni 10 e variamo la dimensione della popolazione da 10 a 100 otteniamo delle soluzioni, in termini di densità media, leggermente migliori (+2.16% per $k = 5$, +1.67% per $k = 10$, +0.70% per $k = 20$), ma a discapito del tempo impiegato, il quale risulta dalle 8.65 alle 10.07 volte più lento. Inoltre, in due occasioni, la densità media riportata nel caso con $h = 100$ è minore del caso con $h = 10$: nel dataset *Students* con $k = 20$ ed $\alpha = 0.2$ all'algoritmo occorrono 203.25 secondi per terminare l'esecuzione con popolazione $h = 100$ rispetto ai 20 secondi impiegati per $h = 10$, portando ad una densità media lievemente peggiore (64.46 contro 64.45); stessa situazione accade nel dataset *Enron* con $k = 20$ ed $\alpha = 0.1$, in cui l'algoritmo ritorna una soluzione peggiore impiegando quasi 9 volte il tempo impiegato dalla variante con dimensioni di popolazione ridotte.

Come era prevedibile, ammettere una sovrapposizione tra gli intervalli porta a soluzioni migliori in termini di densità. A parità di numero di generazioni e di dimensione della popolazione, all'aumentare del valore di α la qualità della soluzione aumenta sempre, come aumenta il tempo d'esecuzione.

Se poniamo il caso con $k = 5$ (come mostrato nella Tabella 3.1) le soluzioni riportate dalla variante con $\alpha = 0.2$ rispetto a quella con $\alpha = 0.05$, mostrano un incremento del 2.74% della densità media, aumentando il tempo d'esecuzione del 5.87%. Osservando, invece, le soluzioni mostrate nella Tabella 3.2, si pùo notare

un aumento del 2.61%, impiegando il 4.69% di tempo in più. Nel caso con $k = 20$ (Tabella 3.3) la densità media, assumendo $\alpha = 0.2$, aumenta del 2.47% rispetto al corrispettivo con $\alpha = 0.05$, rendendolo però più lento del 5.83%.

Da questo si può dimostrare che ammettere una sovrapposizione tra gli intervalli del dominio di tempo in una rete temporale può portare a soluzioni migliori in termini di densità rilevata, senza sacrificare eccessivamente le performance nell'esecuzione.

Confrontiamo ora i risultati ottenuti con i dati forniti in [8] in cui si risolve il problema della ricerca di sottografi densi in reti temporali in cui, però, il dominio di tempo è diviso in intervalli disgiunti. Nella quasi totalità dei casi, ammettendo la sovrapposizione, si ottiene un miglioramento nella densità rilevata. Il miglioramento si dimostra differente al variare dei dataset: il risultato migliore lo ottiene il dataset *Enron*, in cui la soluzione si dimostra maggiore per tutte le combinazioni dei parametri, con un aumento medio del 3.60% fino a un picco del 7.96% nel caso di $\alpha = 0.2$ e $k = 5$, con $g = 10$ & $h = 10$. Se prendiamo, invece, come riferimento il dataset *Facebook*, la soluzione migliora finchè la dimensione della popolazione è piccola. Quando si incrementa la dimensione della popolazione ($h = 100$), la densità media rilevata anzichè migliorare, diminuisce, con un calo che raggiunge il -2.22% nel caso di $\alpha = 0.05$ e $k = 5$ con $g = 5$ & $h = 100$, e mediamente del -1.04%.

In generale, assumendo come limite di sovrapposizione $\alpha = 0.05$, si ottiene un miglioramento della densità, in media, dello 0.53% rispetto alla soluzione in cui non sono ammesse sovrapposizioni; se aumentiamo α rendendolo uguale a 0.1, l'incremento nella qualità della soluzione è pari a 1.37%; infine, se poniamo $\alpha = 0.2$ l'incremento che si ottiene è pari al 3.20%.

Ammettere la sovrapposizione risulta quindi una buona strategia per incrementare la soluzione del problema.

Durante i test effettuati si è constatato che la sovrapposizione tra gli intervalli non raggiungeva mai il suo massimo, se non in rari casi. Questo è dovuto al fatto che nell'inizializzazione e nel rimescolamento genetico del cromosoma è presente una buona dose di casualità, la quale può portare a una riduzione dell'overlap. Un altro motivo può essere dato dal criterio con cui si effettua l'operazione di crossover: in questo lavoro i figli generati tramite crossover sono formati dalle due metà derivate dai cromosomi genitori così come sono, in questo modo le sovrapposizioni dei nuovi cromosomi rimangono identiche a quelle dei genitori rendendole, quindi,

molto dipendenti dalla fase di inizializzazione (controllando sempre che siano valori ammissibili, si veda Sezione 2.3.1). Per questi fattori è stato calcolato che in media in un cromosoma, ad esempio con $k = 5$ e parametro $\alpha = 0.1$, la sovrapposizione totale tra gli intervalli si aggira intorno al 3% rispetto al massimo teorico del 10%. Un'alternativa valida potrebbe essere quella di cercare di massimizzare, ad ogni passaggio, il valore δ; in questo modo si otterrebbe una sovrapposizione più marcata tra gli intervalli e, di conseguenza, una soluzione migliore in termini di densità.

k = 5		*Students*			*Enron*			*Facebook*			*Twitter*		
	α	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**
g=5	*Dens*	26.41	26.56	27.13	42.15	42.51	43.99	14.80	14.79	14.96	23.33	23.66	23.91
h = 10	*Time*	10.42	10.53	10.87	6.60	6.66	6.94	31.51	31.74	32.79	32.05	37.72	40.28
g = 10	*Dens*	26.50	26.74	27.13	42.49	42.90	44.12	14.92	15.08	15.18	23.55	23.74	24.19
h = 10	*Time*	22.15	23.45	23.45	13.59	13.48	13.88	62.49	63.62	66.06	64.06	64.66	66.63
g = 5	*Dens*	26.62	26.87	27.41	42.83	43.74	44.78	15.31	15.43	15.53	23.80	23.94	24.40
h = 100	*Time*	109.16	115.26	119.06	68.20	68.82	70.55	333.43	335.98	345.97	338.07	340.43	349.94
g = 10	*Dens*	26.77	27.03	27.64	43.30	44.03	45.36	15.54	15.56	15.66	23.91	24.09	24.46
h = 100	*Time*	221.66	226.20	233.16	135.53	137.58	141.18	667.33	673.56	691.11	688.38	698.55	718.03

Tabella 3.1: Tabella comparativa dei valori medi di densità e tempi d'esecuzioni per $k = 5$ al variare di α, numero di generazioni e dimensione della popolazione. Il tempo è espresso in secondi.

k = 10		*Students*			*Enron*			*Facebook*			*Twitter*		
	α	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**
g=5	*Dens*	39.37	39.71	40.62	64.11	65.42	66.97	24.41	24.42	24.58	34.63	34.84	35.43
h = 10	*Time*	9.20	9.37	9.75	6.62	6.72	6.88	28.83	29.24	30.32	28.85	29.44	30.56
g = 10	*Dens*	39.54	39.99	40.89	64.54	65.53	67.40	24.78	24.66	25.10	34.89	35.16	35.66
h = 10	*Time*	19.64	19.82	20.49	13.35	13.49	13.98	57.75	58.95	61.06	57.88	59.06	61.34
g = 5	*Dens*	39.90	40.08	40.94	64.57	65.51	67.10	25.02	25.04	25.39	35.14	35.31	35.80
h = 100	*Time*	98.82	101.91	102.54	66.44	67.81	69.50	309.76	315.63	323.29	308.58	312.78	323.59
g = 10	*Dens*	40.19	40.46	41.27	65.33	66.20	67.91	25.41	25.59	25.64	35.45	35.71	36.26
h = 100	*Time*	195.21	195.21	203.55	134.30	136.42	139.34	623.26	633.40	649.95	640.75	642.67	662.17

Tabella 3.2: Tabella comparativa dei valori medi di densità e tempi d'esecuzioni per $k = 10$ al variare di α, numero di generazioni e dimensione della popolazione. Il tempo è espresso in secondi.

k = 20		*Students*			*Enron*			*Facebook*			*Twitter*		
	α	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**	**0.05**	**0.1**	**0.2**
g=5	*Dens*	62.61	63.16	64.21	96.31	97.76	100.71	41.79	41.87	42.20	54.98	55.37	55.89
h = 10	*Time*	9.66	9.83	10.11	7.05	7.25	8.31	27.90	28.54	29.55	28.06	28.35	29.70
g = 10	*Dens*	62.99	63.40	64.46	97.24	98.89	101.51	42.03	42.31	42.47	55.25	55.60	56.32
h = 10	*Time*	19.07	19.43	20.00	14.58	14.75	15.18	55.95	57.33	59.89	56.15	58.44	61.45
g = 5	*Dens*	62.66	63.23	64.27	96.50	97.69	101.33	42.00	42.07	42.56	55.14	55.40	55.97
h = 100	*Time*	98.96	98.60	100.10	70.57	72.33	73.65	294.84	297.26	308.02	291.20	296.49	305.07
g = 10	*Dens*	63.29	63.69	64.45	97.81	98.81	101.85	42.57	42.68	43.28	55.66	56.10	56.81
h = 100	*Time*	191.80	194.72	203.25	142.77	144.86	149.11	591.55	597.03	616.07	601.95	610.80	627.73

Tabella 3.3: Tabella comparativa dei valori medi di densità e tempi d'esecuzioni per $k = 20$ al variare di α, numero di generazioni e dimensione della popolazione. Il tempo è espresso in secondi.

Conclusioni

In questa tesi si è discusso il problema di trovare sottografi densi in reti temporali. Dopo aver introdotto il problema, è stata presentata una variante di tale problema chiamata *k-Densest-Overlapping-Episodes* nel quale si ammette una sovrapposizione tra gli intervalli in cui è suddiviso il dominio di tempo della rete. Sono state mostrate le definizioni utili a delineare il problema ed è stato introdotto un algoritmo che lo risolve basato sugli algoritmi genetici, i quali operano sul dominio di tempo per cercare la segmentazione in intervalli che offrano il miglior risultato, mentre l'algoritmo greedy di Charikar che calcola il sottografo più denso del grafo attivo nell'intervallo dato. Sono stati poi presentati i risultati di tale algoritmo mostrando tempo di esecuzione e qualità delle soluzioni su quattro differenti dataset reali dove viene evidenziato come l'utilizzo di una sovrapposizione, anche se minima, può portare dei benefici nella soluzione.

Risvolti futuri in questo ambito potrebbero essere l'utilizzo di approcci diversi per risolvere questo problema. Per esempio tramite le tecniche di *machine learning* e le reti neurali si potrebbe cercare una soluzione da comparare con i risultati prodotti in questo lavoro, estendendo i test anche a reti temporali di dimensioni maggiori.

Il campo dell'analisi di reti temporali è solo agli inizi ma, data la portata di tale disciplina e visti anche il numero di lavori pubblicati in questo breve periodo, si può prevedere che diventi sempre più argomento centrale nei prossimi anni in diversi campi applicativi, dalla biologia alle scienze economiche e sociali.

Bibliografia

1. Pattillo, J.; Youssef, N.; Butenko, S., 2013. *On clique relaxation models in network analysis.* Eur. J. Oper. Res.2013,226, 9–18.

2. Komusiewicz, C.; Hüffner, F.; Moser, H.; Niedermeier, R., 2009. *Isolation concepts for efficiently enumerating dense subgraphs.* Theor. Comput. Sci. 2009, 410, 3640–3654.

3. Schäfer, A., 2009. *Exact Algorithms for s-Club Finding and Related Problems.* Diplomarbeit (diploma thesis), Friedrich-Schiller-Universität Jena: Jena, Germany.

4. Seidman, S.B., 1983. *Network structure and minimum degree.* Soc. Netw. 1983, 5, 269–287.

5. Goldberg, A.V., 1984. *Finding a Maximum Density Subgraph.* Technical Report. University of California at Berkeley. Berkeley, CA, USA.

6. Chaikar, M., 2000. *Greedy Approximation Algorithms for Finding Dense Components in a Graph.* Stanford University, Stanford, CA, USA.

7. Komusiewicz, C., 2016. *Multivariate algorithmics for finding cohesive subnetworks.* Algorithms 9(1), 21.

8. Castelli, M.; Dondi, R.; Hosseinzadeh, M., 2020. *Genetic Algorithms for Finding Episodes in Temporal Networks*, kes2020.

9. Goldstein, D., 2010. *The dense k subgraph problem.* Master thesis. Open University. Israel.

10. Leeuwen, Jan van, ed. , 1998. *Handbook of Theoretical Computer Science.* Vol. A, Algorithms and complexity. Amsterdam: Elsevier.

11. Rozenshtein, P., Gionis, A., 2019. *Mining temporal networks*, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM. pp. 3225–3226.

12. Rozenshtein, P., Bonchi, F., Gionis, A., Sozio, M., Tatti, N., 2019. *Finding events in temporal networks: Segmentation meets densest subgraph discovery.* Knowledge and Information Systems .

13. Dondi, R., Hosseinzadeh, M.M., 2021. *Dense Sub-networks Discovery in Temporal Networks.* SN COMPUT. SCI. 2, 158. https://doi.org/10.1007/s42979-021-00593-w

14. Kinnear, K. E., 1994. *A Perspective on the Work in this Book.* In K. E. Kinnear (Ed.), Advances in Genetic Programming (pp. 3-17). Cambridge: MIT Press.

15. Mitchell, M., 1996. *An introduction to genetic algorithms.* Complex adaptive systems, MIT press, Cambridge (Mass.).

16. Marcel Danesi, 2006. *Il problema dei ponti di Königsberg di Eulero, in Labirinti, quadrati magici e paradossi logici*, Dedalo, Bari, pp. 89-110.

17. Kenneth Appel, Wolfgang Haken, John Koch, 1977. *Every Planar map is Four Colorable*, Illinois Journal of Mathematics, vol. 21, pp. 439–567.